Postal

Network Working Group
Request for Comments: 94
NIC 5725

2/3/71
E. Harslem
J. Heafner

SOME THOUGHTS ON NETWORK GRAPHICS

PURPOSE

This note states some of our initial reactions to NWG/RFC #86, whose purpose was to provide a basis for discussion and development of Network graphics.

The method of operation described in Note 86 was to interpret data structures to produce graphic order codes for display. This method has proven satisfactory in the past and we favor this approach. The Note 86 proposal is directed toward a particular concept of operation (i.e., minimal graphics terminal connected to computational facilities at remote sites); our remarks embrace extended operations that include smart programs at each end of the connection as well as the minimal terminal.

The proposal in Note 86 should be broadened to include the description of more complex entities and it should be raised to a level of describing more general things. In this note, we first critisize the limitations imposed by the details of Note 86; then suggest some supplimentary ingredients to extend its scope; and lastly, we suggest an alternate approach that reduces Network conversations (where possible) to symbol manipulation rather than gross detail.

COMMENTS ON THE DETAILED RESTRICTIONS OF NOTE 86

The detailed constraints enumerated in Note 86 restrict many interesting features of the Rand display hardware that we consider necessary (from a human factors standpoint) to some current applications. They likewise restrict other nodes whose ARPA-sponsored research is dependent upon the use of sophisticated hardware. For example, the point, vector, and character capability of Note 86 excludes line type mode, intensity control, and many other attractive control operations; the maximum symbol sizes are too small for our large character size; the origin of all of our symbols is specified as the "centroid" of the symbol rather than the lower left corner of a virtual rectangle encompassing the symbol; under mode control for plotting purposes, the beam may not be advanced to the next character position; 7-bit ASCII is insufficient; etc. In short, the five list items of Note 86 are not expressive enough; for example, there is nothing to allow one to position and open a graphic compare "window". The problem was not treated of supplying parameters identifying structure for match, etc. that are not actual display commands.

Perhaps some necessary information gathering (i.e., the display hardware descriptions and characteristics of every node) is preliminary to the generation of a detailed specification. It is important that, without delay, a mechanism be defined for gathering and collating this information in such a way that it doesn't deter progress on Network graphics development.

SOME GENERAL EXTENSIONS TO THE NOTE 86 PROPOSAL

1. Display language capabilities should encompass the union of current and anticipated Network graphics hardware. Our experience in exploring interactive graphics communication techniques for use by researchers and non-programmers indicates that this is not just a "motherhood". The utility of such applications programs depends highly upon incorporating sophisticated graphics hardware. In the absence of those features, some of the programs simply won't be used.
2. The data structure should allow logical as well as pictorial representation of the user's problem. This close coupling of the meaning of a picture with the actual picture is desirable from a processing program's point of view, especially if a user is to interact with the picture. We have found this an efficient way to operate with the GRAIL Project and its derivatives here at Rand. This technique is included in a recently proposed graphics language generated by Bob Anderson (Rand) and Ben Wegbreit (Harvard).
3. Transmit definitions of graphics and then instances of their use. The attempt here is to raise the level of "conversation" between programs (where possible) and to reduce processing overhead. For example, if one wishes to draw lots of resistors, why not graphically define a resistor once and then transmit instances by giving the definition name accompanied by attributes? A typical form of an instance is shown below.

Item Name (position, size, intensity, scaling, labeling, rotation, etc.)

There are many examples of this approach such as the recent work by William Newman (Utah) and many earlier studies at MIT.

4. Partition the display structure for 1) static vs. dynamic information, and 2) context. As opposed to refreshing an entire picture whose domain is the entire screen, we have found it useful to give the processing routine (that wishes to draw a picture) knowledge of only a named rectangular portion of the CRT and an accompanying display structure. With our particular hardware we can then update only the dynamic part of a picture rather than regenerating the entire display structure. Just as important, we can logically assign areas of the CRT to different concurrent processing routines. Coupled with the logical/pictorial representation noted in 2) above, this is a powerful technique. Named partitions also naturally accommodate those applications requiring multiple CRTs.

5. The interpreter could be context-driven thus not restricting its output to a single set of CRT order codes. By providing cataloged descriptions such as the "forms" discussed in Note #83, the interpreter could reconfigure data destined for files, etc., as well as a display. The gain here in terms of adapting to a users' Network needs is large; the price paid in terms of implementing this increment of the interpreter is probably small.

AN ALTERNATE PROPOSAL

Note 86 mentions the case of a terminal at a node with a minimal HOST connected to a remote computationally-oriented node. The data standard, which Note 86 suggests transmitting over the Network is rather gross detail. Also, the standard language is rather inexpressive -- encompassing only a few simple notions.

An alternative approach is to consider the situation of communication between non-minimal nodes (nodes with substantial memory and computing power). Here the Network standard data should be a high-level macro form representing the instances of gross detail with the power to deal with sophisticated graphics devices. That is, the standard language would be rich enough to express all the special features of Network display devices.

This suggestion presents two problems. First, how can a terminal handle commands from a remote program of which its hardware is incapable? The answer is that the remote program to which it is connected is too sophisticated for the terminal -- the connection is invalid. A terminal should normally only connect to a program that addresses no more than its hardware capabilities. This concept allows a standard under which a simple terminal and a simple program can communicate (exactly the proposal of Note 86), yet a sophisticated terminal can talk to a sophisticated program in a high-level language, or it can talk to a simple program, all within the same Network standard.

The second problem is that a minimal host might not have sufficient facilities to translate from a powerful Network standard language into the simple, detailed order codes of its terminals.

When required, the needs of a minimal site would be handled by another Network node providing data reconfiguration services, an essential part of this proposal. The reconfiguration would be done on the basis of "forms" specifying the translation from the Network standard to the specific non-standard data format required by the minimal node (i.e., tailored specifically to its hardware). Whether it would be graphic order codes or some intermediate form would depend on the processing power and requirements of the minimal node.

Fig. 1 shows a schematic diagram of the key elements of such a reconfiguration facility. Fig. 2 shows the use of that facility by a local display handler and its use as an intermediary by two remote nodes requiring different degrees of external data reconfiguration.

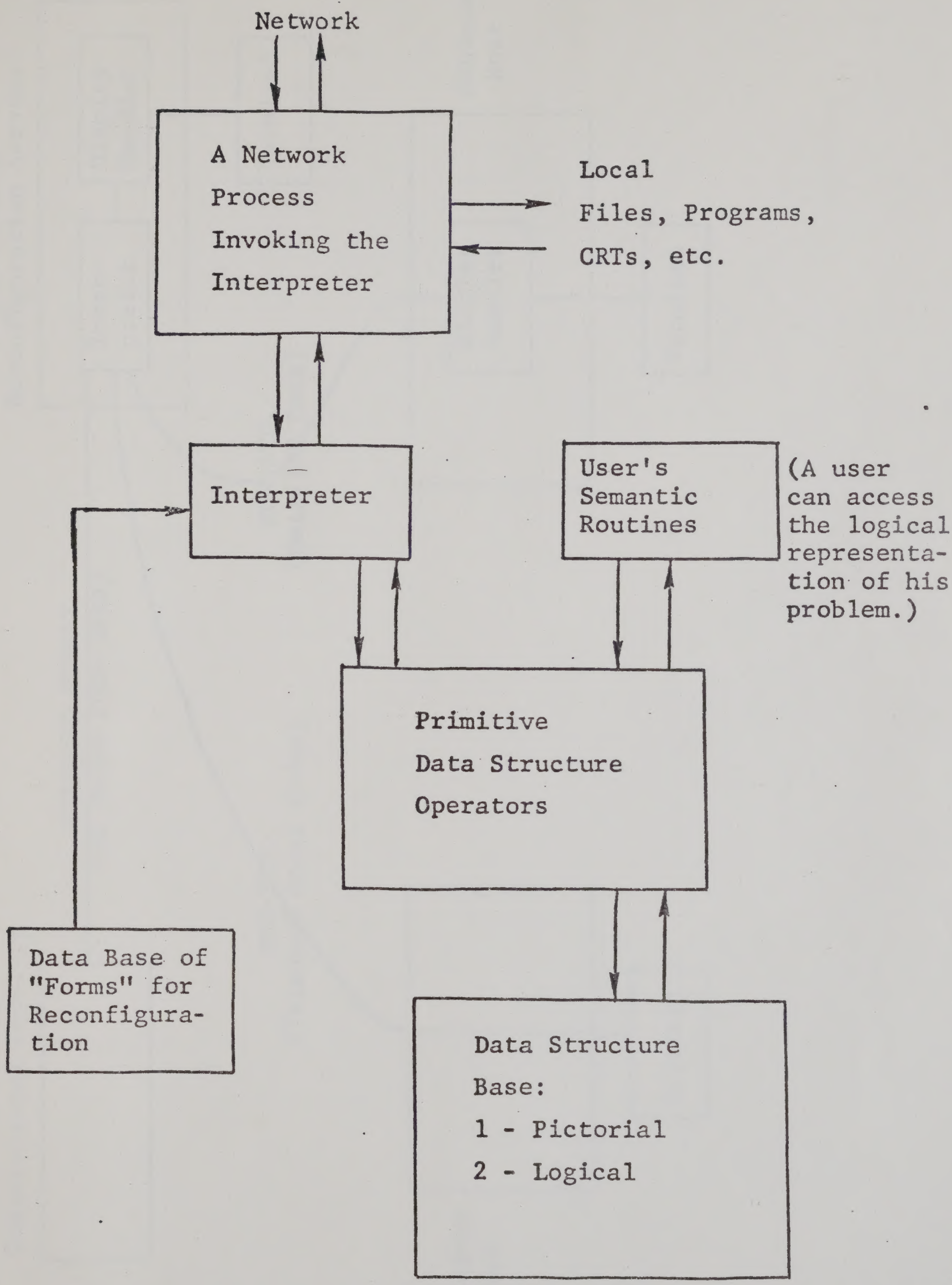
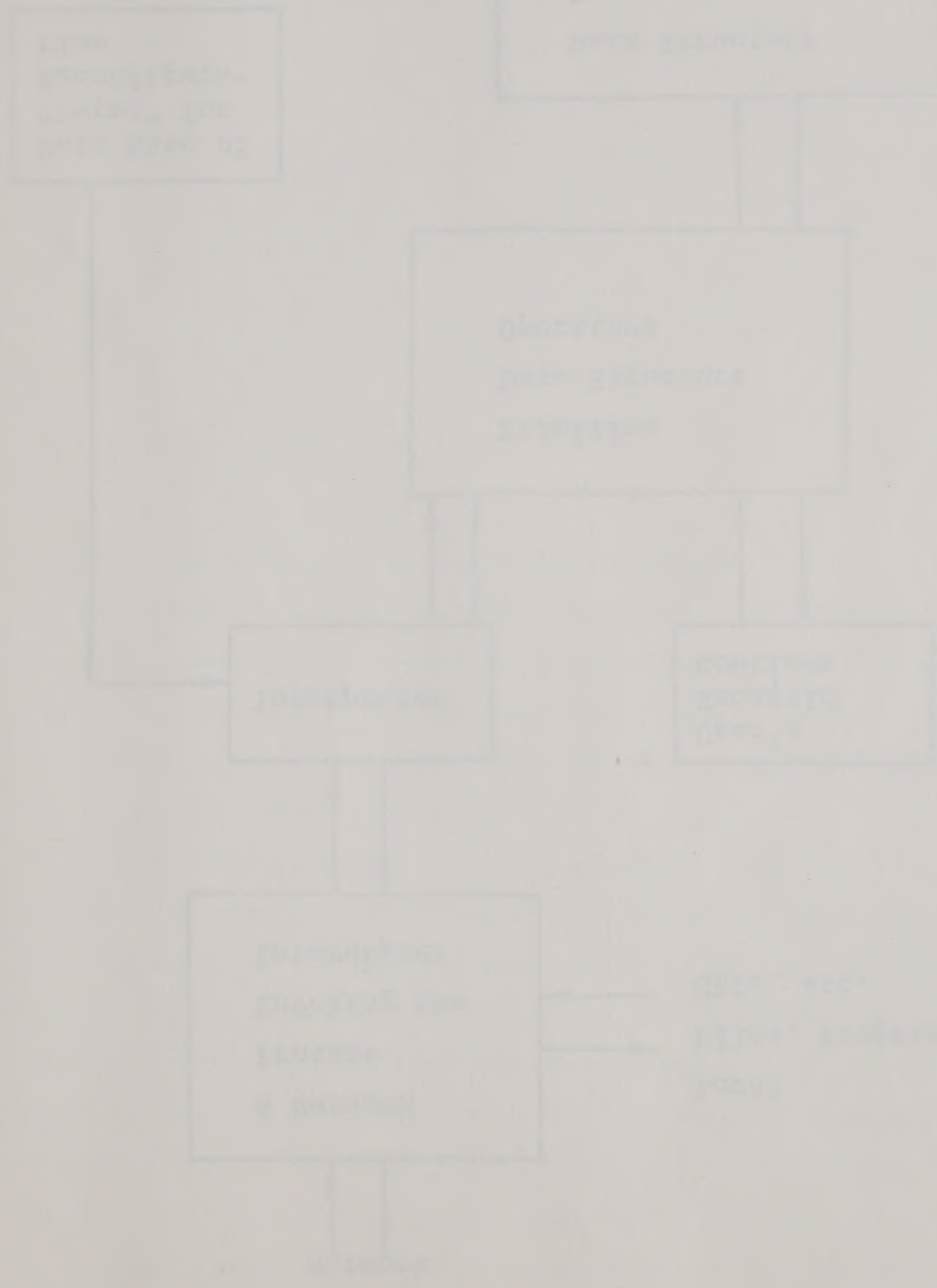

Network
A Network Process Invoking the Interpreter
Local Files, Programs, CRTs, etc.
Interpreter
User's Semantic Routines
(A user can access the logical representation of his problem.)
Primitive Data Structure Operators
Data Base of "Forms" for Reconfiguration
Data Structure Base:
1 - Pictorial
2 - Logical

Fig. 1. Data Reconfiguration Service



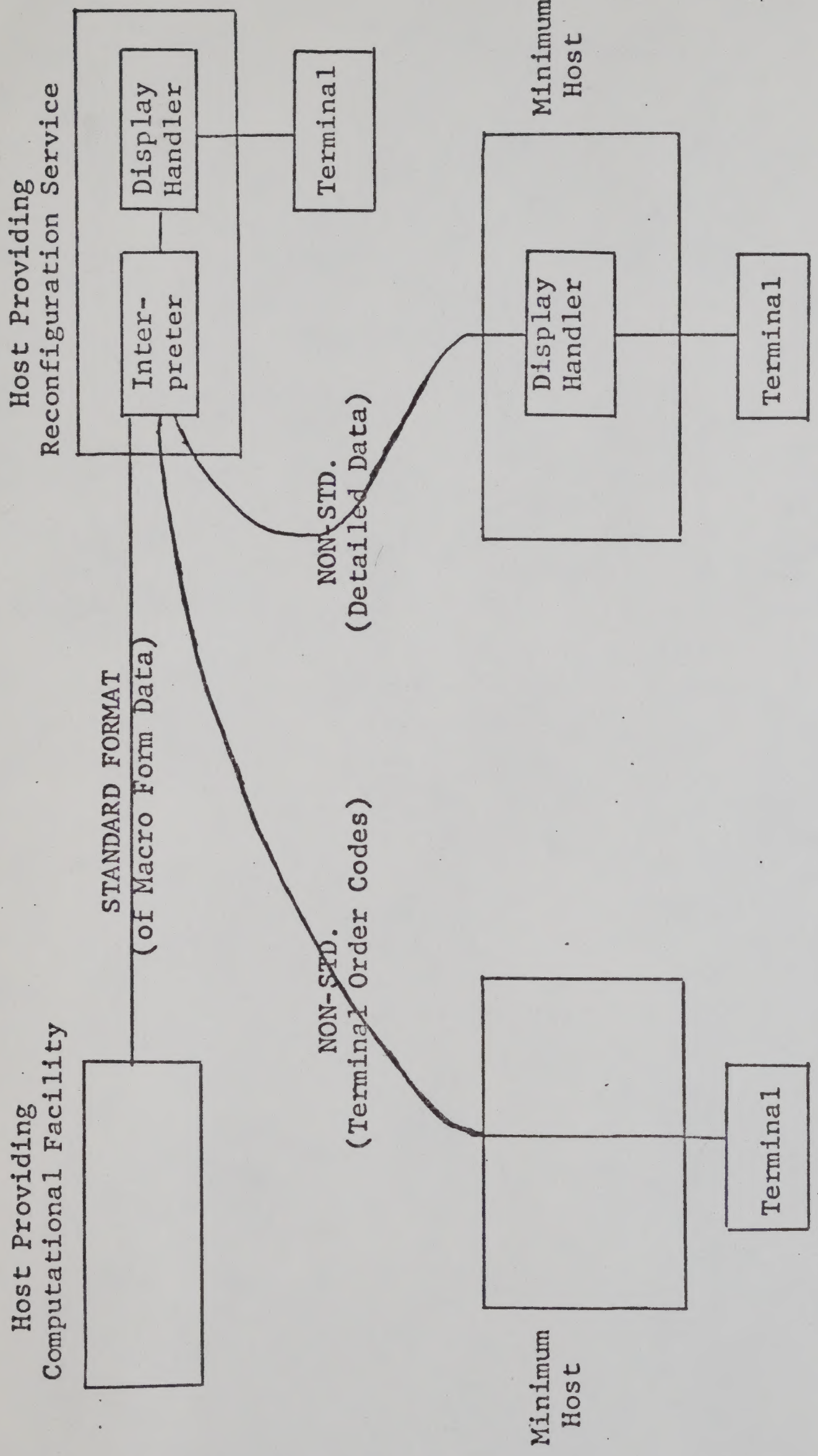


Fig. 2. Use of Data Reconfiguration Service